MP-R EPROM PROGRAMMER USER'S GUIDE

EPROM Availability

We can supply the 2716 EPROMS used in the MP-R. Price is $50.00 retail. Please note that you must not attempt to use anything but a 5Volt power supply EPROM in this system. Texas Instruments 2716 is not this type. No 2708, or 1702 type EPROM can be used. These are all older type, three supply voltage units.

MP-R EPROM Programmer Important Note

Before applying power to and using your MP-R programmer board as called for in the instructions, attach and solder the 0.1 mfd @ 25VDC disc capacitor supplied with the kit across resistor R6. Carefully wrap the leads of the capacitor around those of resistor R6 before soldering. Be sure the capacitor's leads do not contact any adjacent components or PC traces. This capacitor reduces voltage spikes on the 25 volt bus that can damage the EPROM's being programmed.

MP-R EPROM Programmer User's Guide

Introduction

The SWTPC EPROM Programmer and its accompanying software is a plug on option for the SWTPC 6800 Computer System. It is capable of programming any Intel 2716 (5 volt only) pin compatible EPROM. EPROM stands for Erasable Programmable Read Only Memory. Each 2716 is capable of storing 2K bytes of data and may be erased and reprogrammed up to one hundred times. The 2716 is erased by removing the opaque window cover from the top of the package and exposing the device to high level ultraviolet light for a specified amount of time. This amount of time will vary so consult the literature supplied with the ultraviolet light source that you plan to use. After erasing be sure to recover the glass window with the opaque cover. The 2716 can actually be erased by ambient sun and fluorescent light over a long period of time.

The SWTPC MP-A2 processor board on the 6800 Computer System has socket provisions for up to four 2716 pin compatible EPROM's. This gives the MP-A2 board the capability of up to 8K of EPROM. The EPROM's are switch addressable from C000 thru DFFF or from E000 thru FFFF. The C000 thru DFFF range is usually used for PROM BASIC or custom program applications. The E000 thru FFFF range is used for custom monitor or dedicated controller applications and may not be used simultaneously with the MP-A2's 6830 ROM monitor. The MP-A processor board has no provisions for EPROM memory.

EPROM Programmer Hardware and Software

The EPROM programmer hardware consists of a single, 5¼" x 5¼" circuit board which plugs onto one of the unused card slots on the interface bus of the SWTPC 6800 Computer System. The board contains a 24-pin integrated circuit socket where the EPROM to be programmed or verified is inserted. The EPROM may be programmed and verified from the socket on the programming board; however, the program code in the EPROM may not be executed from the EPROM programming board. You may duplicate EPROM's by plugging a programmed EPROM into the socket, having the programmer's software read and store the code, replace the programmed EPROM with an unprogrammed one and program using the stored code.

The software for the EPROM programmer consists of the programmer's monitor (distinct from the processor's monitor), editor and data table and occupies the lower 4K of memory. The monitor gives the user the ability to check, read, program and verify EPROM's. The editor gives the user the ability to load, modify and examine the data which will be stored to EPROM. Both the monitor and the editor occupy the first 2K bytes of memory when loaded into the SWTPC 6800 Computer System. The 2K bytes of data to be stored into the 2716 may be resident in any contiguous 2K byte memory segment. The programmer's software assumes the 2K to 4K segment if no specific addresses are given. This 2K byte block of memory wherever it may be located in memory is called the DATA TABLE. The starting address of this data table is called the BASE ADDRESS. When the EPROM programmer starts transferring code to the EPROM being programmed, the data stored in the BASE ADDRESS is stored in EPROM address 0000. The data stored in the next sequential memory address is stored in EPROM address 0001. The pattern continues all the way up to EPROM address 07FF, the highest address in the 2716 2K byte EPROM. Take note that the addresses within the EPROM's do not correspond with either the addresses of the data table or the actual addresses at which the EPROM will be accessed when installed on the MP-A2 board.

The data within the data table may be loaded by hand using the programmer's editor or may be loaded from a Mikbug® or SWTBUG® or SWTPC CORES assembler ASCII formatted object cassette or paper tape using the Programmer editor's load command. The object tape to be loaded must be contiguous data with only one ORG statement. Multiple ORG statements may only be used if they are followed only by RMB (reserve memory bytes) directives.

PC Board Assembly

NOTE: Since all of the holes on the PC board have been plated thru, it is only necessary to solder the components from the bottom side of the board. The plating provides the electrical connection from the

"BOTTOM" to the "TOP" foil of each hole. Unless otherwise noted it is important that none of the connections be soldered until all of the components of each group have been installed on the board. This makes it much easier to interchange components if a mistake is made during assembly. Be sure to use a low wattage iron (not a gun) with a small tip. Do not use acid core solder or any type of paste flux. We will not guarantee or repair any kit on which either product has been used. Use only the solder supplied with the kit or a 60/40 alloy resin core equivalent. Remember all of the connections are soldered on the bottom side of the board only. The plated-thru holes provide the electrical connection to the top foil.

(✓) Before installing any parts on the circuit board, check both sides of the board over carefully for incomplete etching and foil "bridges" or "breaks". It is unlikely that you will find any; but should there be one especially on the "TOP" side of the board it will be very hard to locate and correct after all of the components have been installed on the board.

(✓) Starting from one end of the circuit board install each of the three, 10-pin Molex female edge connectors along the lower edge of the board. These connectors must be inserted from the "TOP" side of the board and must be pressed down firmly against the circuit board so that each pin extends completely into the holes on the circuit board. Not being careful here will cause the board to either wobble and/or be crooked when plugging it onto the mother board. It is suggested that you solder only the two end pins of each of the three connectors until all have been installed at which time, if everything looks straight and rigid, you should solder the as yet unsoldered pins.

(✓) Insert the small nylon indexing plug into the edge connector pin indicated by the small triangular arrow on the "BOTTOM" side of the circuit board. This prevents the board from being accidently plugged on incorrectly.

(✓) Install all of the resistors on the circuit board. As with all other components unless noted, use the parts list and component layout drawing to locate each part and install from the "TOP" side of the board bending the leads along the "BOTTOM" side of the board and trimming so that 1/16" to 1/8" of wire remains. Solder. Make sure the extended leads of resistor R1 do not inadvertently contact any surrounding printed circuit traces.

(✓) Install all of the capacitors on the circuit board. Be sure to orient electrolytic capacitors C2 and C5 as shown in the component layout drawing. Solder.

(✓) Install the transistors and diodes. These components must be oriented to match the component layout drawing. Solder.

NOTE: Install transistor Q5 so that its metal base is spaced away from the circuit board about 1/16" to prevent the transistor's case from contacting the circuit board's foil. A special nylon spacer may be supplied with the kit for this purpose. If so, use it. Install diode D3 so that its flat side matches with that shown in the component layout drawing.

(✓) Install integrated circuit IC2 on the circuit board. This component must be oriented so its metal face is facing the circuit board and is secured to the circuit board with a #4-40 x 1/4" screw, lockwasher and nut. A heatsink is not used. The three leads of the integrated circuit must be bent down into each of their respective holes. Solder.

(✓) Install integrated circuits IC3, IC4 and IC5 on the circuit board. Do not bend the leads on the back side of the board. Doing so makes it very difficult to remove the integrated circuits should replacement ever be necessary. The semi-circle notch or dot on the end of the package is used for orientation purposes and must match with the outlines shown on the component layout drawing for each of the IC's. Solder.

(✓) Install inductor L1 on the board. Solder.

(✓) Install the 24-pin integrated circuit socket for IC6 on the circuit board. Orient the socket so the handle is adjacent to the top, right corner of the circuit board. Solder the socket with the contacts in the "OPEN" position.

NOTE: MOS integrated circuits are susceptible to damage by static electricity. Although some degree of protection is provided internally within the integrated circuits, their cost demands the utmost in care. Before opening and/or installing any MOS integrated circuits you should ground your body and all metallic tools coming into contact with the leads, thru a 1 M ohm ¼ watt resistor (supplied with the kit). The ground must be an "earth" ground such as a water pipe, and not the circuit board ground. As for the connection to your body, attach a clip lead to your watch or metal ID bracelet. Make absolutely sure you have the 1 Meg ohm resistor connected between you and the "earth" ground, otherwise you will be creating a dangerous shock hazard. Avoid touching the leads of the integrated circuits any more than necessary when installing them, even if you are grounded. On those MOS IC's being soldered in place, the tip of the soldering iron should be grounded as well (separately from your body ground) either with or without a 1 Meg ohm resistor. Most soldering irons having a three prong line cord plug already have a grounded tip. Static electricity should be an important consideration in cold, dry environments. It is less of a problem when it is warm and humid.

( ) Install MOS integrated circuit IC1 following the precautions given in the preceding section. As it is installed, make sure it is down firmly against the board before soldering all of the leads. Do not bend the leads on the back side of the board. Doing so makes it very difficult to remove the integrated circuit should replacement ever be necessary. The "dot" or "notch" on the end of the package is used for orientation purposes and must match with that shown on the component layout drawing for the IC. Solder.

( ) Working from the "TOP" side of the circuit board, fill in all of the feed-thru's with molten solder. The feed-thru's are those unused holes on the board whose internal plating connects the "TOP" and "BOTTOM" circuit connections. Filling these feed-thru's with molten solder guarantees the integrity of the connections and increases the current handling capability.

( ) Now that all of the components have been installed on the board, double check to make sure all have been installed correctly in their proper location.

( ) Check very carefully to make sure that all connections have been soldered. It is very easy to miss some connections when soldering which can really cause some hard to find problems later during checkout. Also look for solder "bridges" and "cold" solder joints which are another common problem.

Since the MP-R circuit board now contains MOS devices, it is susceptible to damage from severe static electrical sources. One should avoid handling the board any more than necessary and when you must, avoid touching or allowing anything to come into contact with any of the conductors on the board.

Parts List - MP-R EPROM Programmer

Resistors

——	R1	4.7K ohm 1/2 watt resistor
——	R2	470 ohm 1/4 " "
——	R3	1 ohm 1/2 " "
——	R4	24.3K ohm precision resistor
——	R5	1.21K " "
——	R6	10K ohm 1/2 watt resistor
——	R7	4.7K ohm 1/4 watt resistor
——	R8	470 ohm " " "
——	R9	470 ohm " " "
——	R10	470 ohm " " "
——	R11	330 ohm " " "

Capacitors

——	C1	150 pfd capacitor
——	C2	*470 mfd electrolytic capacitor
——	C3	0.1 mfd capacitor
——	C4	0.1 mfd capacitor
——	C5	*100 mfd electrolytic capacitor

Semiconductors

——	D1	*1N4003 silicon diode
——	D2	*1N4003 silicon diode
——	D3	*LED
——	Q1	*2N5210 NPN transistor
——	Q2	*2N5210 NPN transistor
——	Q3	*2N4402 PNP transistor
——	Q4	*78L05 5 volt regulator
——	Q5	*SS1122 PNP transistor

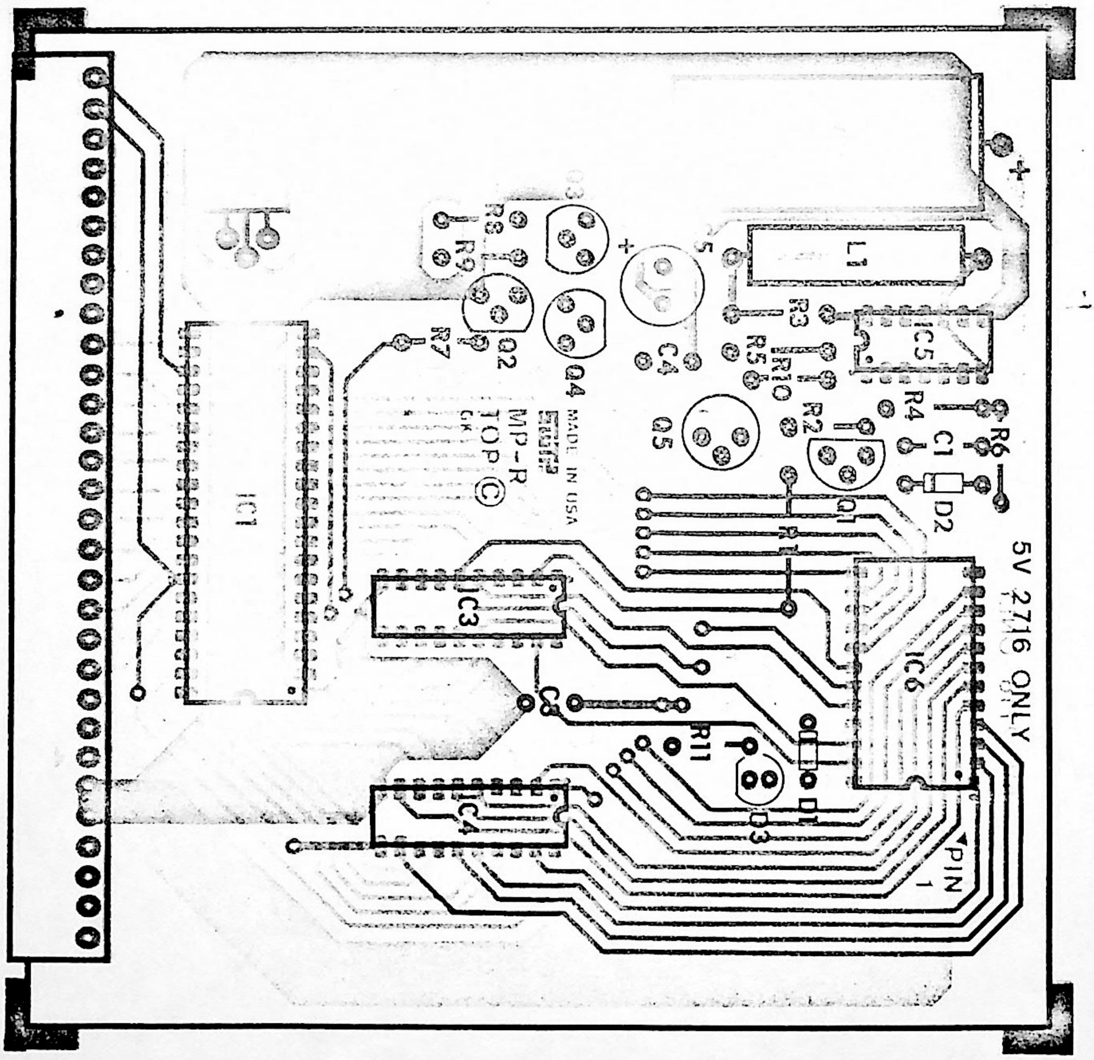
Integrated Circuits

——	IC1	*6820 or 6821 PIA
——	IC2	*7805 5 volt voltage regulator
——	IC3	*74LS273 octal D-flop
——	IC4	*74LS273 octal D-flop
——	IC5	*TL497 switching regulator
——	IC6	2716 (5volt only EPROM and not supplied with kit)

Miscellaneous

——	L1	200uH, 1A inductor

NOTE: All components flagged with a * must be orientated as shown in the component layout drawing.


5V 2716 ONLY
PIN 1
L1
IC5
IC6
IC1
IC3
IC4
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
C1
C2
C4
C5
D1
D2
D3
Q1
Q2
Q3
Q4
Q5
+
MADE IN USA
MP-R
TOP ©

Using the Programmer

The EPROM programmer may be plugged onto any one of the unused I/O card positions on the back of the SWTPC 6800 Computer System. Upon executing the programmer's software the computer will respond with a request for the I/O port number onto which the EPROM programmer board is plugged. The correct I/O number should be entered at that time. Don't forget that the I/O's are numbered from 0 thru 7 left to right. The correct number for each I/O position is printed on the back of the mother board adjacent the interface.

```
SWT PRMPRG VER 1.0


I/O PORT #3
ARE YOU SURE? Y
```

EPROM Programmer's Monitor

Once the EPROM programmer software is loaded into the system via cassette or paper tape and the program is started by typing G for "Go to the User Program", the system comes up in the EPROM programmer's monitor which is distinct from the system's operating system ROM monitor. The programmer's monitor prompts with a > and is capable of responding with any one of seven two letter commands. You need only type the first two characters of each command. The computer prints out the remainder for you. The BASE ADDRESS is initially set at 0800 and must be reset before the WRITE command if you desire something other than this. Typing a "Control X" at any time will transfer program control back to the monitor. There is no provision for backspace.

The EPROM programmer monitor's commands are as follows:

UNPGMMED CHECK
READ
PRINT
EDIT
WRITE TO PROM
VERIFY
EXIT

Command Summary

UNPGMMED CHECK—The unprogrammed check command verifies that every byte in the EPROM over the address range specified is unprogrammed (=FF). If any bytes are not set to FF as they should be, the user is notified, otherwise the message UNPGMMED is printed. Be sure to check each erased EPROM before you program it to make sure that all bytes have been reset to FF. The address range over which the EPROM is checked is selectable since the EPROM may be selectively programmed. It is possible to program any segment of the EPROM without disturbing non-selected addresses. Example:

```
>UNPGMMED CHECK
BGN PROM ADDR=0000
END PROM ADDR=07FF
UNPGMMED
```

or

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

or

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
```

```
♦ERROR♦
INCON: 0000 DATA=FF PROM=00
CONTINUE? Y

♦ERROR♦
INCON: 0001 DATA=FF PROM=00
CONTINUE? N

>
```

In this command as with all others, responding with a carriage return to the "BGN ADDR=" question will default the programmer to the entire EPROM address range (0000 thru 07FF). Responding with a carriage return to the "END ADDR=" question will default the programmer to the single byte entered for the "BGN ADDR=" question.

READ—The READ command is used to transfer information from an already programmed EPROM into the data table so that either part or all of the data may be edited and/or written into another EPROM. The READ command automatically resets the BASE ADDRESS to 0800 and asks for the address range to be transferred from the EPROM being read to the DATA TABLE. Executing a READ with no EPROM installed in the socket will fill the DATA TABLE with 00's over the address range specified.

```
>READ
BGN PROM ADDR=<CR>   CARRIAGE RETURN

>
```

PRINT—The PRINT command prints the DATA TABLE over the EPROM address range specified. The data is printed with the EPROM address followed by sixteen bytes of data, repeated until the entire specified address range has been printed. You may prematurely exit the PRINT sequence by repeatedly striking any key until a ">" is printed. Example:

```
>PRINT
BGN ADDR=0000
END ADDR=001F

<ADDR>  <DATA>

0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

>
```

EDIT—The EDIT command transfers control to the programmer's editor which is described later in this manual. Example:

```
>EDIT

TYPE B,M,L, OR X:

>
```

WRITE TO EPROM—The WRITE TO EPROM command transfers the data stored in the DATA TABLE to the EPROM over the EPROM address range specified. Upon completion the EPROM is automatically verified for accuracy. If a discrepancy is found the program prints BYTE WILL NOT PROGRAM followed by the location of the discrepancy. The total program and verification time varies up to two minutes. **Do not** RESET or power down while writing to the EPROM. Example:

```
>WRITE TO PROM
BGN PROM ADDR=0000
END PROM ADDR=0100
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

or

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING

◆ERROR◆
BYTE WILL NOT PGM
0000 DATA=FF PROM=00
CONTINUE? Y

◆ERROR◆
BYTE WILL NOT PGM
0001 DATA=FF PROM=00
CONTINUE? N

>.
```

VERIFY—The VERIFY command verifies that the data stored in the EPROM over the address range specified is the same as the data stored in the DATA TABLE. This function is automatically executed as part of the WRITE command. If no discrepancies are found, the message VERIFIED GOOD is printed. Example:

```
>VERIFY
BGN PROM ADDR=(CR)
VERIFIED GOOD

>
```

or

```
>VERIFY
BGN PROM ADDR=0000
END PROM ADDR=0100

◆ERROR◆
INCON: 0000 DATA=FF PROM=00
CONTINUE? Y

◆ERROR◆
INCON: 0001 DATA=FF PROM=00
CONTINUE? N

>
```

EXIT—The EXIT command returns control to the MIKBUG® or SWTBUG® monitor. PRMPRG may be re-entered, however, without losing data stored in the DATA TABLE simply by typing G for Go to User Program. Exiting and then restarting the program will reset the BASE ADDRESS to the default value of 0800. Example:

```
>EXIT

$
```

EPROM Programmer's Editor

When the command EDIT is typed when in the EPROM programmer's monitor, control is transferred to the EPROM programmer's editor. From the Editor, the user may modify or create a DATA TABLE any-where within computer memory except for the lower 2K bytes (0000 - 07FF) where the EPROM Programmer software resides. Once the Editor is entered, the following message is typed on the screen:

TYPE B, M, L or X:

These characters are defined below:

- B —change BASE ADDRESS of the DATA TABLE
- M —modify the DATA TABLE (change data or create DATA TABLE)
- L —Read object file from assembler or monitor punch routine
- X —exit Editor and return to programming monitor

These command characters are defined as follows:

B —When B is typed, the old BASE ADDRESS of the DATA TABLE is printed. The user may keep that address by typing a carriage return, or may change it by typing a 4-digit hex address which will become the new BASE ADDRESS.

M —When the M command is typed, the computer will respond with:

NEW TABLE?

If a Y for yes is entered the BASE ADDRESS is set to the default value of 0800 and all data within the table is set to $FF. If a N for no is entered, the BASE ADDRESS is left at whatever value it was previously set to. If it was not previously set by the user, it will be at the default value of 0800. The computer will then respond with:

BGN PROM ADDR=

You should at this time type in the 4-digit hex address of the EPROM memory address you wish to examine and/or change. The address may be from 0000 thru 07FF inclusive. Attempting to access an address out of this range will return you back to the editor. The computer will line feed, return, echo the address, display its contents and a space.

At this point the user has the option of advancing, either forward or backward, to the next memory location, or changing the data stored at the displayed address and advancing to the next location or of exiting the M function.

To display the next sequential address and data, type a line feed. Any leading spaces that are entered will be ignored by the memory change function.

To display the next sequential address going backward from the present location, a ↑ should be entered.

To change the data stored at the displayed location, enter the new data, either with or without a leading space. If a non-hex value, such as a 3Q is entered the data will remain unchanged a "?" will be printed and the address will be repeated. If the data is unable to be changed (write protected memory, etc.) a ? will be output and the address will be repeated.

To exit the Memory Examine and Change function, type a carriage return. Example:

```
TYPE B,M,L, OR X: M
NEW TABLE? N
BGN PROM ADDR=0000
0000 FF 01
0001 FF  02
0002 FF   03
0003 FF    04
0004 FF ^
0003 04 ^
0002 03 ^
0001 02 ^
0000 01 (CR)
TYPE B,M,L, OR X:
```

L —The L or LOAD command is used to load a SWTPC CORES editor/assembler, MIKBUG® or SWTBUG® ASCII formatted tape into the DATA TABLE. The BASE ADDRESS of the DATA TABLE is automatically reset to the default address of 0800 and all object data is loaded in sequentially from this address regardless of the assigned object address. For example, an assembler object tape ORG'ed to be located sequentially from 0100 thru 0300 inclusive would be loaded into the data table from 0800 thru 0A00 inclusive. Since programs to be stored to ROM may only be located from C000 thru FFFF on the MP-A2 board, this 0100 thru 0300 program would have to be written with address independent code (no extended addressing) to actually function at an address location other than which it was ORG'ed at. Typical EPROM programs will have to be ORG'ed at low addresses so that they may be written and debugged in RAM memory. Once debugged, the program may be ORG'ed and reassembled for high memory where it may be burned into EPROM. If the anticipated EPROM program is to be stored in the C000 thru DFFF region, extra MP-M or MP-8M memory boards may be modified as per the MP-A2 instructions to be addressed in high memory so that programs may initially be assembled and tested in high memory from the start without the need to go back and re-ORG and reassemble.

Since the LOAD command loads all data sequentially starting from the BASE ADDRESS, only one ORG statement is allowed in assembler generated code unless subsequent ORG statements are followed by RMB (reserve memory byte) directives only. All object data is stored sequentially from the 0800 BASE ADDRESS. If you are trying to store an 8K byte program into four sequential 2K byte EPROM's. The object code data for these EPROM's would be stored sequentially from memory location 0800 thru 27FF. This is a total of 10K bytes of RAM memory. To program these EPROM's we would first erase (if not already so) the four EPROM's we intend to use. We would then insert the EPROM programming board into one of the unused card positions on the back of the SWTPC 6800 Computer System and load and execute the PRMPRG program as follows:

```
SWT PRMPRG VER 1.0


I/O PORT #3
ARE YOU SURE? Y
```

We then enter the programmer's editor and use the **L** command to load the 8K byte object tape. When we enter the L, the computer responds with the BLOCK #. If we have enough memory to store all of the object code to be stored to EPROM we enter 0. This causes the computer to load object data sequentially from the very first byte on the tape. If we have less than enough memory, say for example only 4K (2K of which is for the PRMPRG program itself) we must load the object data to memory, 2K blocks at a time. Specifying BLOCK #1 loads the second 2K block, and so on; all the way up to block #9. The disadvantage here is that the object tape must be repeatedly loaded to reposition the data in the DATA TABLE. In the following example let's assume that we have 10K of memory so that all of the object code may be read and stored in one pass-

```
>EDIT

TYPE B,M,L, OR X: L      Insert object tape in recorder and PLAY.
BLOCK #0                 Read data will be printed on the terminal.
TYPE B,M,L, OR X: X
                        Insert the first EPROM in the programmer's socket
>UNPGMMED CHECK
BGN FROM ADDR=(CR)
UNPGMMED
```

Continued next page

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=0800  1000
TYPE B,M,L, OR X: X
```

Reset BASE ADDRESS for the next 2K block

Remove the first and
Insert the second EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=1000  1800
TYPE B,M,L, OR X: X
```

Reset BASE ADDRESS for the next 2K block

Remove the second and
Insert the third EPROM in the Programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=1800  2000
TYPE B,M,L, OR X: X
```

Reset BASE ADDRESS for the next 2K block

Remove the third and
Insert the fourth EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>
```

Remove the fourth EPROM

In the following example, we load the same program one block at a time as if we have only 4K of memory in our computer system:

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #0
TYPE B,M,L, OR X: X
```

Insert object tape in recorder and PLAY.
Read data will be printed on the terminal
Insert the first EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: L
BLOCK #1
TYPE B,M,L, OR X: X
```

Rewind and re-insert the object tape and PLAY

Remove the first and insert the second EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED


>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: L
BLOCK #2
TYPE B,M,L, OR X: X
```

Rewind and re-insert the object tape and PLAY

Remove the second and insert the third EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

Continued next page

```
>EDIT

TYPE B,M,L, OR X: L
```

Rewind and re-insert the object tape and PLAY

```
BLOCK #3
TYPE B,M,L, OR X: X
```

Remove the third and insert the fourth EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

Remove the fourth EPROM

```
>
```

X —The X command is used to EXIT the Editor and return to the programmer's monitor. Example:

```
>EDIT

TYPE B,M,L, OR X: X

>
```

A Word of Warning

The timing and voltage switching for the MP-R EPROM Programmer are under software control. The control portion of the program resides in the lower 2K bytes of memory and it is very important that this memory segment as well as the rest of the processor operate flawlessly. If there is a problem it is possible to permanently damage the EPROM being programmed. It is suggested therefore that you run the memory diagnositics on your system prior to using the EPROM programmer just as a precaution. It is also suggested that you familiarize yourself with the operation of the programmer's software by operating the board without a EPROM installed until you feel comfortable with the software. Do not RESET or power down the system while you are WRITING to EPROM.

Testing the Programmer

It is very important that you check to make sure that the 25.0 VDC programming voltage is correct before attempting to program an EPROM. Clip the leads of a DC voltmeter across electrolytic capacitor C2. Use the polarity markings of the capacitor for proper connection. Load the PRMPRG software and give it a READ command over the entire address range with no EPROM installed in the socket. This will fill the data table with zeros (00's). Now execute a WRITE command over the entire address range. The voltmeter should read between 24.0 and 26.0 VDC. Don't forget to take into account the inaccuracies in your voltmeter. If the measured voltage is too high, you should increase the value of resistor R5 approximately 5 ohms for each 0.1 volt difference. If the voltage is too low, you should increase the value of resistor R4 approximately 100 ohms for each 0.1 volt difference. The ideal voltage is 25.0 VDC. Having the voltage too high (above 26.0 volts) can destroy the EPROM being programmed. Recheck the voltage after making any changes.

How It Works

Interfacing from the 6800 bus to the EPROM and its programming circuitry is done thru PIA IC1. Since more data lines than were available on one PIA are necessary, latches IC3 and IC4 are used to provide the extra outputs. Data is loaded into the latches by strobing the clock line of the latches with the CA2 and CB2 line of the PIA. Since no DC voltage higher than 12 volts is available on the SWTPC 6800 Computer System bus, it is necessary to use an integrated swtiching regulator to up the unregulated 7-8VDC bus voltage to the 25.0 VDC required for programming. Transistors Q1 and Q5 control the voltage to the switching regulator so the switcher may be disabled when not in use. Since the EPROM is usually installed and removed while computer system power is applied, transistors Q2 and Q3 control the EPROM +5 VDC regulator so that power is not applied to the socket during installation and removal.

The programming sequence is under total software control and starts by setting the CS input high which is the normal state during programming. The PD/PGM input is set low and the 25.0 VDC power supply is switched on and applied to the Vpp input. The selected address and data to be stored are applied to address and data inputs of the EPROM and the PD/PGM input is pulsed high for 45 to 55 milliseconds which programs the byte. This process is repeated for all bytes in which at least one bit must be set to zero. Only those bytes which must be set to something other than FF's are actually programmed. The unprogrammed (erase) state of each byte in the 2716 EPROM is FF.

The read sequence sets the PD/PGM input and CS inputs low, sets the selected byte address up on the EPROM, and inputs the EPROM data thru the B side of the PIA.

Handling the 2716 EPROM

The 2716 EPROM is a MOS device and is susceptable to damage from static electricity. It is shipped and should be stored with its leads impressed into a conductive material (usually conductive foam) for maximum protection. To transfer the 2716 from the conductive foam to the programmer's socket, place one hand on the conductive foam and grasp all of the leads of the IC with the other hand. Carefully remove the IC. While grasping all of the leads with the fingers of one hand grab the computer chassis with the other. This brings both to the same voltage potential level. While contacting the computer's chassis install the IC into the programmer's zero force socket. Make sure, of course, that the socket is open and ready to accept the IC and that the IC is oriented correctly. The dot or notch on the end of the IC's package must match with the PIN 1 indicator on the PC board. After insertion, double check for proper orientation and close the latching socket.

The MP-R's LED diode D3 is lit whenever +5 VDC power is applied to the socket. Never install or remove the EPROM to the socket while this diode is lit. The light is normally off unless data within the EPROM is being accessed.

After programming, follow the same precautions when transferring the 2716 to the conductive foam or MP-A2 board. Place one hand on the computer's chassis and open the socket. While contacting the computer's chassis, remove the IC with the fingers in contact with as many of the IC's pins as possible. While grasping the IC with one hand place the other on either the conductive foam or MP-A2 board and carefully insert the IC. If you are inserting the IC into the MP-A2 board, be sure to orient the IC as indicated on the MP-A2 board or its component layout drawing. Remember also that the lid of the 2716 should be covered after programming to prevent ambient light from erasing the IC over a long period of time.

Erasing 2716 EPROM's

To erase the 2716 EPROM it must be exposed to an ultraviolet light source emiting wavelengths shorter than 4000 Angstroms. Although fluorescent lights and sunlight emit wavelengths in this region, the amount of time required for erasure would vary from months to years. Those special UV sources emitting wavelengths required for erasing are hazardous to the skin and eyes and must be handled with care. Typical erasing time using a short wavelength UV lamp varies from twenty to sixty minutes. EPROM's should not be erased while power is applied since current paths exist that effectively cancel the energy being provided by the UV light.

At the time of this writing only one manufacturer to our knowledge has been advertising an EPROM eraser for the hobbyist. This is:

Ultra-Violet Products Inc.
5100 Walnut Grove Ave.
San Gabriel, CA 91778

The erasers are sold thru dealers and retail for $59.50. If you would rather build your own there is an article on just that on page 91 of the January 1977 issue of BYTE magazine.

Error Messages

When either user errors or execution errors occur, the user is notified thru error messages. Some errors require user response to continue execution. For example, if an inconsistancy is found using the verify routine, the following message is printed:

```
*ERROR*
INCON: 00F6 DATA=C1 PROM=D1
CONTINUE?
```

The user may acknowledge the error and continue by typing a **Y** for yes or may abort and return to the monitor by typing a **N** for no. The following error messages may be generated:

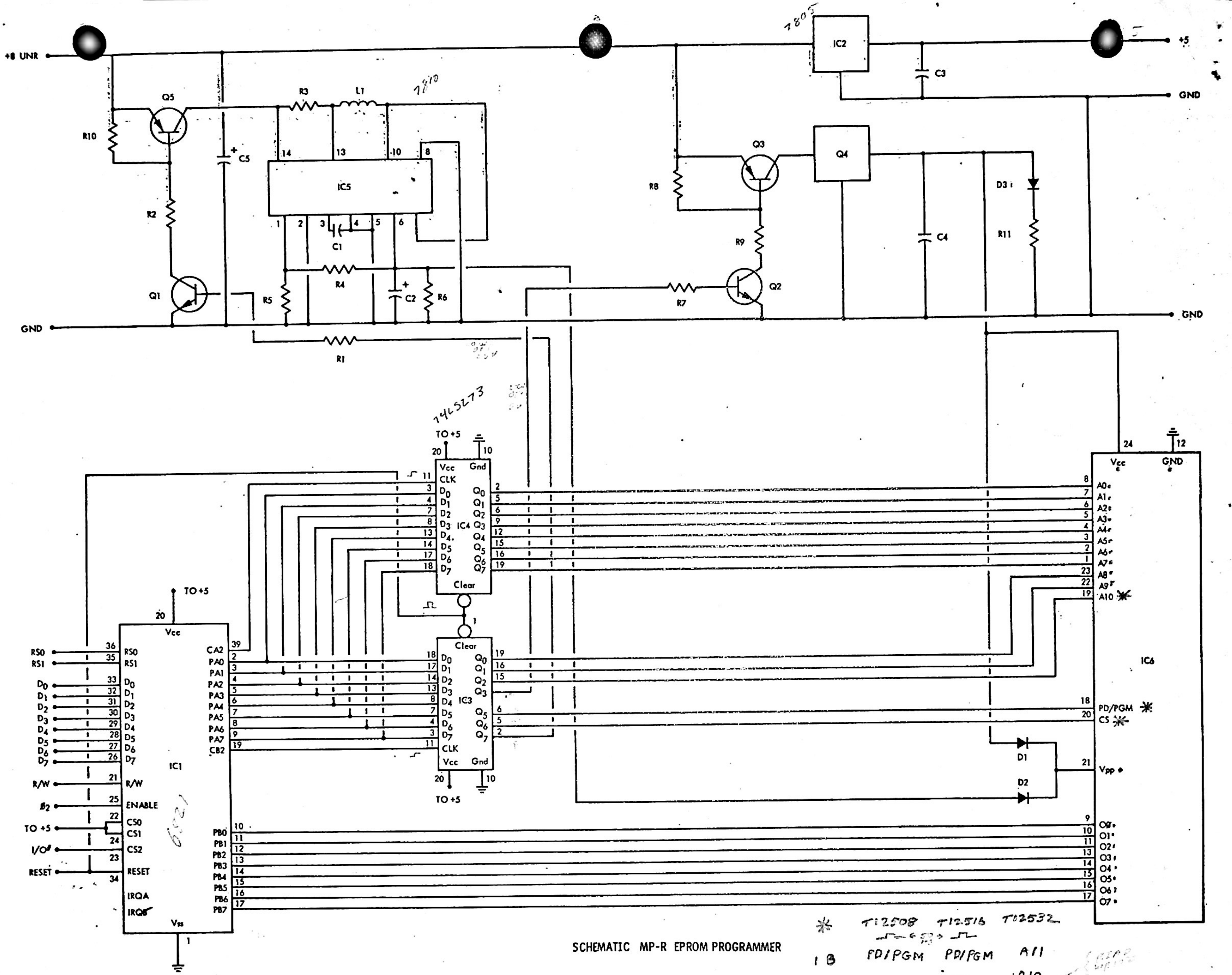
OUT OF RANGE–The specified address limits are larger than 07FF, which is the capacity of the 2716 EPROM.

BYTE WILL NOT PGM–The address in the EPROM printed after this message will not program correctly.

END<BGN–Specified addresses are out of order.

INCON:– A data inconsistency has been found.

MIKBUG® is a registered trademark of Motorola Inc.
SWTBUG® is a registered trademark of Southwest Technical Products Corp.


+8 UNR
+5
GND
GND
GND
IC1
IC2
IC3
IC4
IC5
IC6
Q1
Q2
Q3
Q4
Q5
R1
R2
R3
R4
R5
R6
R7
R8
R9
R10
R11
C1
C2
C3
C4
C5
L1
D1
D2
D3
7805
7810
74LS273
6821
TO +5
Vcc
Gnd
CLK
Clear
RS0
RS1
D_0
D_1
D_2
D_3
D_4
D_5
D_6
D_7
R/W
ϕ_2
ENABLE
CS0
CS1
CS2
I/O
RESET
IRQA
IRQB
Vss
CA2
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
CB2
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
PD/PGM
CS
Vpp
O0
O1
O2
O3
O4
O5
O6
O7

SCHEMATIC MP-R EPROM PROGRAMMER

*	TI2508	TI2516	TI2532
18	PD/PGM	PD/PGM	A11
19	NC	A10	A10 ← 50ms
20	$\overline{CS}$	$\overline{CS}$	$\overline{PD}$/PGM

Using the Programmer

The EPROM programmer may be plugged onto any one of the unused I/O card positions on the back of the SWTPC 6800 Computer System. Upon executing the programmer's software the computer will respond with a request for the I/O port number onto which the EPROM programmer board is plugged. The correct I/O number should be entered at that time. Don't forget that the I/O's are numbered from 0 thru 7 left to right. The correct number for each I/O position is printed on the back of the mother board adjacent the interface.

```
SWT PRMPRG VER 1.0


I/O PORT #3
ARE YOU SURE? Y
```

EPROM Programmer's Monitor

Once the EPROM programmer software is loaded into the system via cassette or paper tape and the program is started by typing G for "Go to the User Program", the system comes up in the EPROM programmer's monitor which is distinct from the system's operating system ROM monitor. The programmer's monitor prompts with a > and is capable of responding with any one of seven two letter commands. You need only type the first two characters of each command. The computer prints out the remainder for you. The BASE ADDRESS is initially set at 0800 and must be reset before the WRITE command if you desire something other than this. Typing a "Control X" at any time will transfer program control back to the monitor. There is no provision for backspace.

The EPROM programmer monitor's commands are as follows:

UNPGMMED CHECK
READ
PRINT
EDIT
WRITE TO PROM
VERIFY
EXIT

Command Summary

UNPGMMED CHECK—The unprogrammed check command verifies that every byte in the EPROM over the address range specified is unprogrammed (=FF). If any bytes are not set to FF as they should be, the user is notified, otherwise the message UNPGMMED is printed. Be sure to check each erased EPROM before you program it to make sure that all bytes have been reset to FF. The address range over which the EPROM is checked is selectable since the EPROM may be selectively programmed. It is possible to program any segment of the EPROM without disturbing non-selected addresses. Example:

```
>UNPGMMED CHECK
BGN PROM ADDR=0000
END PROM ADDR=07FF
UNPGMMED
```

or

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

or

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
```

```
*ERROR*
INCON: 0000 DATA=FF PROM=00
CONTINUE? Y

*ERROR*
INCON: 0001 DATA=FF PROM=00
CONTINUE? N

>
```

In this command as with all others, responding with a carriage return to the "BGN ADDR=" question will default the programmer to the entire EPROM address range (0000 thru 07FF). Responding with a carriage return to the "END ADDR=" question will default the programmer to the single byte entered for the "BGN ADDR=" question.

READ—The READ command is used to transfer information from an already programmed EPROM into the data table so that either part or all of the data may be edited and/or written into another EPROM. The READ command automatically resets the BASE ADDRESS to 0800 and asks for the address range to be transferred from the EPROM being read to the DATA TABLE. Executing a READ with no EPROM installed in the socket will fill the DATA TABLE with 00's over the address range specified.

```
>READ
BGN PROM ADDR=(CR)   CARRIAGE RETURN

>
```

PRINT—The PRINT command prints the DATA TABLE over the EPROM address range specified. The data is printed with the EPROM address followed by sixteen bytes of data, repeated until the entire specified address range has been printed. You may prematurely exit the PRINT sequence by repeatedly striking any key until a ">" is printed. Example:

```
>PRINT
BGN ADDR=0000
END ADDR=001F

<ADDR>  <DATA>

0000  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
0010  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00

>
```

EDIT—The EDIT command transfers control to the programmer's editor which is described later in this manual. Example:

```
>EDIT

TYPE B,M,L, OR X:

>
```

WRITE TO EPROM—The WRITE TO EPROM command transfers the data stored in the DATA TABLE to the EPROM over the EPROM address range specified. Upon completion the EPROM is automatically verified for accuracy. If a discrepancy is found the program prints BYTE WILL NOT PROGRAM followed by the location of the discrepancy. The total program and verification time varies up to two minutes. **Do not** RESET or power down while writing to the EPROM. Example:

```
>WRITE TO PROM
BGN PROM ADDR=0000
END PROM ADDR=0100
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

or

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING

*ERROR*
BYTE WILL NOT PGM
0000 DATA=FF PROM=00
CONTINUE? Y

*ERROR*
BYTE WILL NOT PGM
0001 DATA=FF PROM=00
CONTINUE? N

>
```

VERIFY—The VERIFY command verifies that the data stored in the EPROM over the address range specified is the same as the data stored in the DATA TABLE. This function is automatically executed as part of the WRITE command. If no discrepancies are found, the message VERIFIED GOOD is printed. Example:

```
>VERIFY
BGN PROM ADDR=(CR)
VERIFIED GOOD

>
```

or

```
>VERIFY
BGN PROM ADDR=0000
END PROM ADDR=0100

*ERROR*
INCON: 0000 DATA=FF PROM=00
CONTINUE? Y

*ERROR*
INCON: 0001 DATA=FF PROM=00
CONTINUE? N

>
```

EXIT—The EXIT command returns control to the MIKBUG® or SWTBUG® monitor. PRMPRG may be re-entered, however, without losing data stored in the DATA TABLE simply by typing G for Go to User Program. Exiting and then restarting the program will reset the BASE ADDRESS to the default value of 0800. Example:

```
>EXIT

$
```

EPROM Programmer's Editor

When the command EDIT is typed when in the EPROM programmer's monitor, control is transferred to the EPROM programmer's editor. From the Editor, the user may modify or create a DATA TABLE any-where within computer memory except for the lower 2K bytes (0000 - 07FF) where the EPROM Programmer software resides. Once the Editor is entered, the following message is typed on the screen:

TYPE B, M, L or X:

These characters are defined below:

B —change BASE ADDRESS of the DATA TABLE
M —modify the DATA TABLE (change data or create DATA TABLE)
L —Read object file from assembler or monitor punch routine
X —exit Editor and return to programming monitor

These command characters are defined as follows:

B —When B is typed, the old BASE ADDRESS of the DATA TABLE is printed. The user may keep that address by typing a carriage return, or may change it by typing a 4-digit hex address which will become the new BASE ADDRESS.

M —When the M command is typed, the computer will respond with:

NEW TABLE?

If a Y for yes is entered the BASE ADDRESS is set to the default value of 0800 and all data within the table is set to $FF. If a N for no is entered, the BASE ADDRESS is left at whatever value it was previously set to. If it was not previously set by the user, it will be at the default value of 0800. The computer will then respond with:

BGN PROM ADDR=

You should at this time type in the 4-digit hex address of the EPROM memory address you wish to examine and/or change. The address may be from 0000 thru 07FF inclusive. Attempting to access an address out of this range will return you back to the editor. The computer will line feed, return, echo the address, display its contents and a space.

At this point the user has the option of advancing, either forward or backward, to the next memory location, or changing the data stored at the displayed address and advancing to the next location or of exiting the M function.

To display the next sequential address and data, type a line feed. Any leading spaces that are entered will be ignored by the memory change function.

To display the next sequential address going backward from the present location, a ↑ should be entered.

To change the data stored at the displayed location, enter the new data, either with or without a leading space. If a non-hex value, such as a 3Q is entered the data will remain unchanged a "?" will be printed and the address will be repeated. If the data is unable to be changed (write protected memory, etc.) a ? will be output and the address will be repeated.

To exit the Memory Examine and Change function, type a carriage return. Example:

```
TYPE B,M,L, OR X: M
NEW TABLE? N
BGN PROM ADDR=0000
0000 FF 01
0001 FF  02
0002 FF   03
0003 FF    04
0004 FF ^
0003 04 ^
0002 03 ^
0001 02 ^
0000 01 (CR)
TYPE B,M,L, OR X:
```

L —The L or LOAD command is used to load a SWTPC CORES editor/assembler, MIKBUG® or SWTBUG® ASCII formatted tape into the DATA TABLE. The BASE ADDRESS of the DATA TABLE is automatically reset to the default address of 0800 and all object data is loaded in sequentially from this address regardless of the assigned object address. For example, an assembler object tape ORG'ed to be located sequentially from 0100 thru 0300 inclusive would be loaded into the data table from 0800 thru 0A00 inclusive. Since programs to be stored to ROM may only be located from C000 thru FFFF on the MP-A2 board, this 0100 thru 0300 program would have to be written with address independent code (no extended addressing) to actually function at an address location other than which it was ORG'ed at. Typical EPROM programs will have to be ORG'ed at low addresses so that they may be written and debugged in RAM memory. Once debugged, the program may be ORG'ed and reassembled for high memory where it may be burned into EPROM. If the anticipated EPROM program is to be stored in the C000 thru DFFF region, extra MP-M or MP-8M memory boards may be modified as per the MP-A2 instructions to be addressed in high memory so that programs may initially be assembled and tested in high memory from the start without the need to go back and re-ORG and reassemble.

Since the LOAD command loads all data sequentially starting from the BASE ADDRESS, only one ORG statement is allowed in assembler generated code unless subsequent ORG statements are followed by RMB (reserve memory byte) directives only. All object data is stored sequentially from the 0800 BASE ADDRESS. If you are trying to store an 8K byte program into four sequential 2K byte EPROM's. The object code data for these EPROM's would be stored sequentially from memory location 0800 thru 27FF. This is a total of 10K bytes of RAM memory. To program these EPROM's we would first erase (if not already so) the four EPROM's we intend to use. We would then insert the EPROM programming board into one of the unused card positions on the back of the SWTPC 6800 Computer System and load and execute the PRMPRG program as follows:

```
SWT PRMPRG VER 1.0


I/O PORT #3
ARE YOU SURE? Y
```

We then enter the programmer's editor and use the L command to load the 8K byte object tape. When we enter the L, the computer responds with the BLOCK #. If we have enough memory to store all of the object code to be stored to EPROM we enter 0. This causes the computer to load object data sequentially from the very first byte on the tape. If we have less than enough memory, say for example only 4K (2K of which is for the PRMPRG program itself) we must load the object data to memory, 2K blocks at a time. Specifying BLOCK #1 loads the second 2K block, and so on; all the way up to block #9. The disadvantage here is that the object tape must be repeatedly loaded to reposition the data in the DATA TABLE. In the following example let's assume that we have 10K of memory so that all of the object code may be read and stored in one pass-

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #0
TYPE B,M,L, OR X: X
```

Insert object tape in recorder and PLAY.
Read data will be printed on the terminal.

Insert the first EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED
```

Continued next page

```
>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=0800  1000
TYPE B,M,L, OR X: X
```

Reset BASE ADDRESS for the next 2K block

Remove the first and
Insert the second EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=1000  1800
TYPE B,M,L, OR X: X
```

Reset BASE ADDRESS for the next 2K block

Remove the second and
Insert the third EPROM in the Programmer's socket.

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: B
BASE ADDR=1800  2000
TYPE B,M,L, OR X: X
```

Reset BASE ADDRESS for the next 2K block

Remove the third and
Insert the fourth EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>
```

Remove the fourth EPROM

In the following example, we load the same program one block at a time as if we have only 4K of memory in our computer system:

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #0
TYPE B,M,L, OR X: X
```

Insert object tape in recorder and PLAY.
Read data will be printed on the terminal

Insert the first EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: L
BLOCK #1
TYPE B,M,L, OR X: X
```

Rewind and re-insert the object tape and PLAY

Remove the first and insert the second EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED


>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD

>EDIT

TYPE B,M,L, OR X: L
BLOCK #2
TYPE B,M,L, OR X: X
```

Rewind and re-insert the object tape and PLAY

Remove the second and insert the third EPROM in the programmer's socket

```
>UNPGMMED CHECK
BGN PROM ADDR=(CR)
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
```

Continued next page

```
>EDIT

TYPE B,M,L, OR X: L
BLOCK #3                    Rewind and re-insert the object tape and PLAY
TYPE B,M,L, OR X: X

->UNPGMMED CHECK            Remove the third and insert the fourth EPROM
BGN PROM ADDR=(CR)          in the programmer's socket
UNPGMMED

>WRITE TO PROM
BGN PROM ADDR=(CR)
...PROGRAMMING
...VERIFYING
PGMMING COMPLETED
VERIFIED GOOD
>                           Remove the fourth EPROM
```

X —The X command is used to EXIT the Editor and return to the programmer's monitor. Example:

```
>EDIT

TYPE B,M,L, OR X: X
>
```

A Word of Warning

The timing and voltage switching for the MP-R EPROM Programmer are under software control. The control portion of the program resides in the lower 2K bytes of memory and it is very important that this memory segment as well as the rest of the processor operate flawlessly. If there is a problem it is possible to permanently damage the EPROM being programmed. It is suggested therefore that you run the memory diagnositics on your system prior to using the EPROM programmer just as a precaution. It is also suggested that you familiarize yourself with the operation of the programmer's software by operating the board without a EPROM installed until you feel comfortable with the software. Do not RESET or power down the system while you are WRITING to EPROM.

Testing the Programmer

It is very important that you check to make sure that the 25.0 VDC programming voltage is correct before attempting to program an EPROM. Clip the leads of a DC voltmeter across electrolytic capacitor C2. Use the polarity markings of the capacitor for proper connection. Load the PRMPRG software and give it a READ command over the entire address range with no EPROM installed in the socket. This will fill the data table with zeros (00's). Now execute a WRITE command over the entire address range. The voltmeter should read between 24.0 and 26.0 VDC. Don't forget to take into account the inaccuracies in your voltmeter. If the measured voltage is too high, you should increase the value of resistor R5 approximately 5 ohms for each 0.1 volt difference. If the voltage is too low, you should increase the value of resistor R4 approximately 100 ohms for each 0.1 volt difference. The ideal voltage is 25.0 VDC. Having the voltage too high (above 26.0 volts) can destroy the EPROM being programmed. Recheck the voltage after making any changes.

How It Works

Interfacing from the 6800 bus to the EPROM and its programming circuitry is done thru PIA IC1. Since more data lines than were available on one PIA are necessary, latches IC3 and IC4 are used to provide the extra outputs. Data is loaded into the latches by strobing the clock line of the latches with the CA2 and CB2 line of the PIA. Since no DC voltage higher than 12 volts is available on the SWTPC 6800 Computer System bus, it is necessary to use an integrated swtiching regulator to up the unregulated 7-8VDC bus voltage to the 25.0 VDC required for programming. Transistors Q1 and Q5 control the voltage to the switching regulator so the switcher may be disabled when not in use. Since the EPROM is usually installed and removed while computer system power is applied, transistors Q2 and Q3 control the EPROM +5 VDC regulator so that power is not applied to the socket during installation and removal.

The programming sequence is under total software control and starts by setting the CS input high which is the normal state during programming. The PD/PGM input is set low and the 25.0 VDC power supply is switched on and applied to the Vpp input. The selected address and data to be stored are applied to address and data inputs of the EPROM and the PD/PGM input is pulsed high for 45 to 55 milliseconds which programs the byte. This process is repeated for all bytes in which at least one bit must be set to zero. Only those bytes which must be set to something other than FF's are actually programmed. The unprogrammed (erase) state of each byte in the 2716 EPROM is FF.

The read sequence sets the PD/PGM input and CS inputs low, sets the selected byte address up on the EPROM, and inputs the EPROM data thru the B side of the PIA.

Handling the 2716 EPROM

The 2716 EPROM is a MOS device and is susceptable to damage from static electricity. It is shipped and should be stored with its leads impressed into a conductive material (usually conductive foam) for maximum protection. To transfer the 2716 from the conductive foam to the programmer's socket, place one hand on the conductive foam and grasp all of the leads of the IC with the other hand. Carefully remove the IC. While grasping all of the leads with the fingers of one hand grab the computer chassis with the other. This brings both to the same voltage potential level. While contacting the computer's chassis install the IC into the programmer's zero force socket. Make sure, of course, that the socket is open and ready to accept the IC and that the IC is oriented correctly. The dot or notch on the end of the IC's package must match with the PIN 1 indicator on the PC board. After insertion, double check for proper orientation and close the latching socket.

The MP-R's LED diode D3 is lit whenever +5 VDC power is applied to the socket. Never install or remove the EPROM to the socket while this diode is lit. The light is normally off unless data within the EPROM is being accessed.

After programming, follow the same precautions when transferring the 2716 to the conductive foam or MP-A2 board. Place one hand on the computer's chassis and open the socket. While contacting the computer's chassis, remove the IC with the fingers in contact with as many of the IC's pins as possible. While grasping the IC with one hand place the other on either the conductive foam or MP-A2 board and carefully insert the IC. If you are inserting the IC into the MP-A2 board, be sure to orient the IC as indicated on the MP-A2 board or its component layout drawing. Remember also that the lid of the 2716 should be covered after programming to prevent ambient light from erasing the IC over a long period of time.

Erasing 2716 EPROM's

To erase the 2716 EPROM it must be exposed to an ultraviolet light source emiting wavelengths shorter than 4000 Angstroms. Although fluorescent lights and sunlight emit wavelengths in this region, the amount of time required for erasure would vary from months to years. Those special UV sources emitting wavelengths required for erasing are hazardous to the skin and eyes and must be handled with care. Typical erasing time using a short wavelength UV lamp varies from twenty to sixty minutes. EPROM's should not be erased while power is applied since current paths exist that effectively cancel the energy being provided by the UV light.

At the time of this writing only one manufacturer to our knowledge has been advertising an EPROM eraser for the hobbyist. This is:

Ultra-Violet Products Inc.
5100 Walnut Grove Ave.
San Gabriel, CA 91778

The erasers are sold thru dealers and retail for $59.50. If you would rather build your own there is an article on just that on page 91 of the January 1977 issue of BYTE magazine.

Error Messages

When either user errors or execution errors occur, the user is notified thru error messages. Some errors require user response to continue execution. For example, if an inconsistancy is found using the verify routine, the following message is printed:

```
*ERROR*
INCON: 00F6 DATA=C1 PROM=D1
CONTINUE?
```

The user may acknowledge the error and continue by typing a Y for yes or may abort and return to the monitor by typing a N for no. The following error messages may be generated:

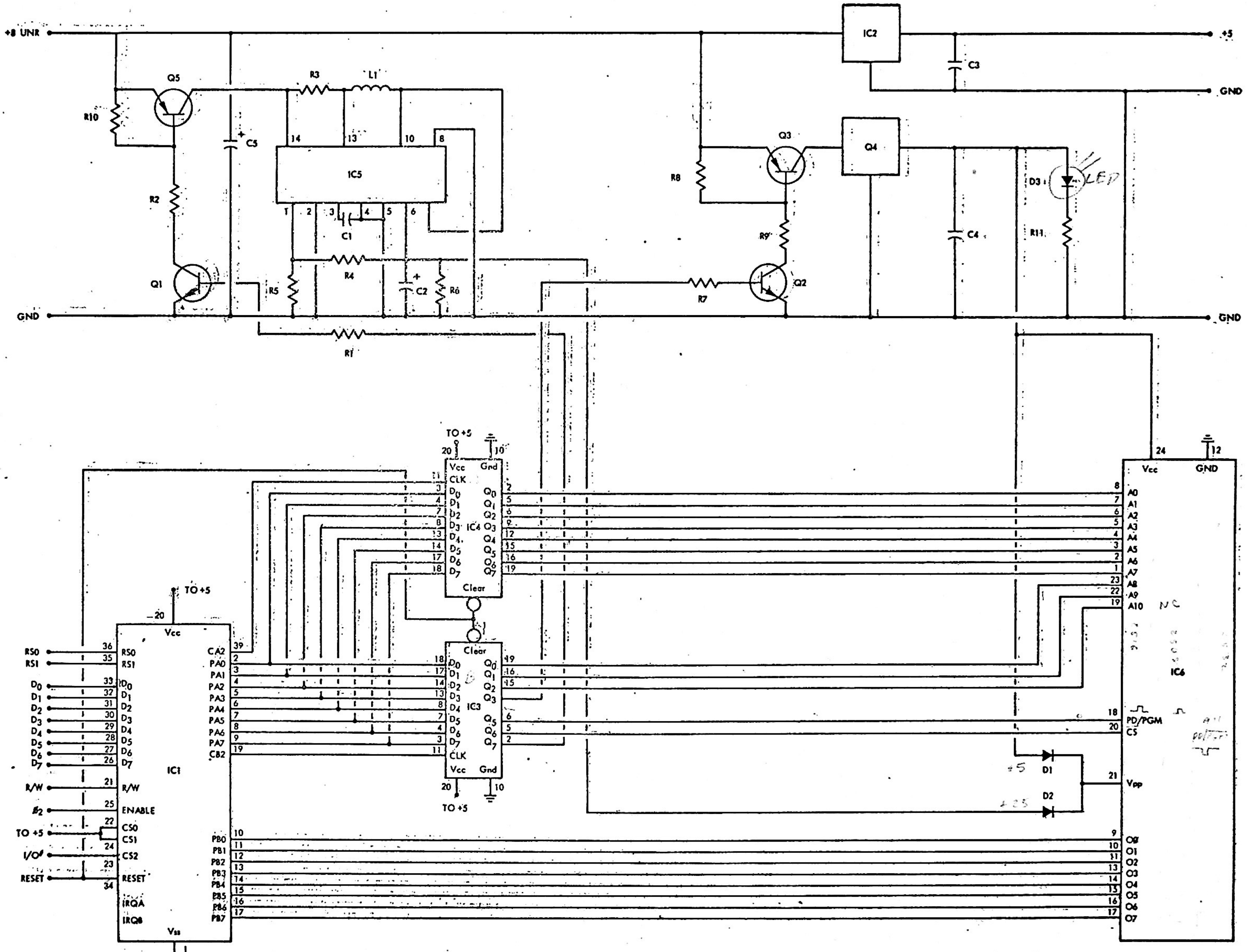
OUT OF RANGE–The specified address limits are larger than 07FF, which is the capacity of the 2716 EPROM.

BYTE WILL NOT PGM–The address in the EPROM printed after this message will not program correctly.

END<BGN–Specified addresses are out of order.

INCON:– A data inconsistency has been found.

MIKBUG® is a registered trademark of Motorola Inc.
SWTBUG® is a registered trademark of Southwest Technical Products Corp.


+8 UNR
GND
+5
GND
GND
Q5
R10
C5
R3
L1
IC5
14
13
10
8
1
2
3
4
5
6
C1
R4
R5
C2
R6
R2
Q1
R1
IC2
C3
Q4
Q3
R8
R9
Q2
R7
C4
D3
LED
R11
TO +5
Vcc
Gnd
CLK
D0
D1
D2
D3
D4
D5
D6
D7
IC4
Q0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Clear
IC3
RS0
RS1
R/W
ENABLE
Ø2
CS0
CS1
CS2
I/O Ø
RESET
IRQA
IRQB
Vss
IC1
CA2
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
CB2
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
IC6
Vcc
GND
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
NC
PD/PGM
CS
Vpp
D1
D2
+5
+25
O0
O1
O2
O3
O4
O5
O6
O7

SCHEMATIC MP-R EPROM PROGRAMMER